PCI Express IO Virtualization and IO Sharing Specification

Revision 0.3

October 13, 2005

REVISION	REVISION HISTORY	DATE
0.1	Initial requirements draft	04/22/05
0.3	Draft requirements for general membership review	10/13/05

The PCI-SIG disclaims all warranties and liability for the use of this document and the information contained herein and assumes no responsibility for any errors that may appear in this document, nor does the PCI-SIG make a commitment to update the information contained herein.

Contact the PCI-SIG office to obtain the latest revision of the specification.

Questions regarding this document or membership in the PCI-SIG may be forwarded to:

Membership Services
http://www.pcisig.com
E-mail: administration@pcisig.com
Phone: 503-291-2569
800-433-5177 (USA Only)
Fax: 503-297-1090

Technical Support
techsupp@pcisig.com

DISCLAIMER

This document is provided "as is" with no warranties whatsoever, including any warranty of merchantability, non-infringement, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification, or sample. The PCI-SIG disclaims all liability for infringement of proprietary rights, relating to use of information in this specification. No license, express or implied, by estoppels or otherwise, to any intellectual property rights is granted herein.

All product names are trademarks, registered trademarks, or service marks of their respective owners.

Copyright © 2005 PCI-SIG

Contents

PREFACE.....
OBJECTIVE OF THE SPECIFICATION.....
DOCUMENT ORGANIZATION.....
DOCUMENTATION CONVENTIONS.....
TERMS AND ABBREVIATIONS
REFERENCE DOCUMENTS

1. INTRODUCTION..... 13

2. IO VIRTUALIZATION ARCHITECTURAL OVERVIEW..... 14

2.1. DEDICATED ENDPOINTS..... 14
2.2. ENDPOINT SHARING VIA SOFTWARE INTERMEDIARY 15
2.3. ADDRESS TRANSLATION AND PROTECTION TABLE..... 16
2.4. NATIVE IOV 17
2.5. MULTI-ROOT TOPOLOGY..... 20
2.6. MRA PCI MANAGER OPERATING PRINCIPLES 22

3. NATIVE IOV – PROTOCOL CHANGES..... 24

4. NATIVE IOV – IOV SEMANTICS..... 25

5. NATIVE IOV INITIALIZATION / CONFIGURATION 26

5.1. CONFIGURATION MECHANISMS 26
5.2. CONFIGURATION TOPOLOGY..... 26
5.3. CONFIGURATION MECHANISMS 27
5.4. CONFIGURATION TRANSACTION RULES..... 27
5.5. CONFIGURATION SPACE..... 27
5.6. CONFIGURATION SPACE STRUCTURE - PHYSICAL MODE 28
5.7. CONFIGURATION SPACE STRUCTURE – VE MODE 28
5.8. VE CREATION..... 28
5.9. VE MODIFICATION 28
5.10. VE DESTRUCTION..... 28

6. NATIVE IOV - EVENT MANAGEMENT..... 29

6.1. VE RESET 29
6.2. VE HOT-PLUG 29
6.3. VE HOT-REMOVE 29

7. NATIVE IOV – ERROR REPORTING..... 30

7.1. ERROR REPORTING MECHANISMS..... 30

8. NATIVE IOV – POWER MANAGEMENT..... 31

9. MULTI-ROOT IOV SEMANTICS..... 32

10. MULTI-ROOT PROTOCOL CHANGES 33

11. MULTI-ROOT INITIALIZATION / CONFIGURATION 34

11.1. CONFIGURATION TOPOLOGY 34
11.2. CONFIGURATION MECHANISMS 34
11.3. CONFIGURATION TRANSACTION RULES 35
11.4. CONFIGURATION SPACE 35
11.5. TRUSTED CONFIGURATION SPACE 35
11.6. ELECTING A PCIM TO MANAGE THE IOV FABRIC 35
11.7. ENFORCING PCIM ELECTION 35
11.8. TRANSFER OF PCIM ELECTION 35
11.9. ESTABLISH AN ERROR COLLECTOR 35
11.10. ASSOCIATION OF VIRTUAL ENTITY TO AN RC 35
11.11. RESET OF AN ENDPOINT ASSOCIATED WITH AN RC 36
11.12. DISASSOCIATION OF AN ENDPOINT FROM AN RC 36
11.13. MONITORING AN IOV FABRIC 36

12. NATIVE IOV - EVENT MANAGEMENT 37

12.1. MRA ENDPOINT RESET 37
12.2. MRA ENDPOINT HOT-PLUG 37
12.3. MRA ENDPOINT HOT-REMOVE 37

13. MULTI-ROOT- ERROR REPORTING 38

13.1. ERROR REPORTING MECHANISMS 38

14. MULTI-ROOT - POWER MANAGEMENT 39

Figures

Tables

Preface

System virtualization creates multiple, Virtual Systems within a platform. Each Virtual System requires its own dedicated IO resources. As a result, in many physical configurations, the number of PCI IO slots available on a client or server will be insufficient to provide each Virtual System with its own dedicated PCI IO Endpoint. The need for PCI IO Endpoint sharing increases in proportion to the number of Virtual Systems. Even where a sufficient number of physical PCI IO Endpoints could be installed, the sharing of PCI IO Endpoints is desirable, because it allows Virtual Systems to share an underutilized Endpoint, thereby reducing overall system costs.

PCIe IO Virtualization also enables IO load balancing and bandwidth management. If a Virtual System needs more bandwidth or connectivity, then more physical Endpoint capacity can be associated with that Virtual System. If a Virtual System begins to consume more IO resources than it should, its consumption can be throttled back down to a preset level. These capabilities can be used by workload management personnel or automation software to manage PCIe IO Virtualization resources.

Objective of the Specification

This specification describes the IO Virtualization capabilities of the PCI Express architecture. It includes the requirements for Endpoints, Switches and PCIe RC / RP to implement IO virtualization over a flexible set of topologies. The goal is to enable interoperability between components Endpoints from different vendors that support the functionality defined within this specification. This specification is intended as an enhancement to the PCIe architecture spanning multiple market segments; Clients, Servers Embedded and Communication Endpoints.

Document Organization

The specification is organized to present two main functions Native IO Virtualization in single and multi-root topologies. As the definition evolves, other companion documents may be published to further enhance standard support of IO virtualization for PCIe Endpoints, Switches and RC / RP. The specification contains the technical details of the architecture, protocol, Link layer, physical layer, and software interface.

Documentation Conventions

Capitalization

Some terms are capitalized to distinguish their definition in the context of this document from their common English meaning. Words not capitalized have their common English meaning. When terms such as "memory write" or "memory read" appear completely in lower case, they include all transactions of that type.
Register names and the names of fields and bits in registers and headers are presented with the first letter capitalized and the remainder in lower case.

Numbers and Number Bases

Hexadecimal numbers are written with a lower case "h" suffix, e.g., 0FFFFh and 80h.
Hexadecimal numbers larger than four digits are represented with a space dividing each group of four digits, as in 1E FFFF FFFFh. Binary numbers are written with a lower case "b" suffix, e.g., 1001b and 10b. Binary numbers larger than four digits are written with a space dividing each group of four digits, as in 1000 0101 0010b.
All other numbers are decimal.

Reference Information

Reference information is provided in various places to assist the reader and does not represent a requirement of this document. Such references are indicated by the abbreviation "(ref)." For example, in some places, a clock that is specified to have a minimum period of 400 ps also includes the reference information maximum clock frequency of "2.5 GHz (ref)." Requirements of other specifications also appear in various places throughout this document and are marked as reference information. Every effort has been made to guarantee that this information accurately reflects the referenced document; however, in case of a discrepancy, the original document takes precedence.

Implementation Notes

Implementation Notes should not be considered to be part of this specification. They are included for clarification and illustration only. Implementation Notes within this document are enclosed in a box and set apart from other text.

Terms and Abbreviations

Address Translation and Protection Table (ATPT)	The data structure(s) accessed by a Translation Agent to determine the mapping of un-translated DMA addresses into translated addresses. The data structure may contain fields that indicate the protection attributes of the translation entry. (Refer to the ATC specification for updates to the definition.)
Configuration Time VI Involvement	Refers to the operations performed by a VI on behalf of a hardware or software agent on a VE to associate it with a System Image.
Intermediary based PCI IOV	An IOV mechanism that uses a Virtualization Intermediary.
IO Virtualization	The capability for a single IO unit to be used by more than one System Image.
IOV	IO Virtualization
IOV Enabled PCIe Component	A PCIe RC, Switch, Bridge, or Endpoint that supports the required IOV extensions defined in this specification. Since MRA extensions are a superset of IOV extensions, an MRA enabled component is also an IOV enabled component.
IOV RC IOV Switch IOV Bridge IOV Endpoint	See IOV Enabled PCIe Component.
Guest Physical Address	The address used by a System Image to access system memory.
MR - Multi-Root Topology	A PCIe topology that interconnects more than one root through multi-root aware Switches.
MRA Enabled PCIe Component Multi-Root Aware Component	A PCIe RC, Switch, Bridge, or Endpoint that supports the required multi-root IOV extensions defined in this specification. MRA components are used to provide each PCIe root complex with its own virtual, PCI 32 or 64-bit Memory, IO and Configuration Space.
MRA RC, MRA Switch MRA Bridge MRA Endpoint	See MRA Enabled Component.
MRA VE	Virtual Endpoint (VE) in an MRA endpoint.
Native IOV	An IOV mechanism where a physical PCI Endpoint supports one or more VEs. Each VE is assigned to an SI using configuration operations that are issued by a VI. Mechanisms exist so that DMA transfers, MMIO accesses and interrupt operations can be performed directly between an SI and the VE, without Run-Time VI Involvement. The mechanism may require Run-Time VI Involvement for some operations (e.g. Configuration Space accesses).

PCI Bus Memory Address	Refers to the address portion of a PCI-* Memory Transaction
PCI Manager (PCIM)	Software or firmware that discovers the PCIe fabric. It also initializes MRA Components such that each PCIe Root complex has its own Virtual Hierarchy.
PCIe	PCI Express
PCIe RC	The PCIe Root Complex, including any Root Ports exported by that RC as defined in the PCIe documentation suite. In a virtual system, the PCIe RC may be virtualized by any combination of hardware and software.
Physical Address	The address used by the system memory controller to access system memory.
RC	Root Complex
RP	Root Port
Run-Time VI Involvement	Refers to the intervention by a VI on operations performed between an SI and a physical PCI Endpoint (versus a VE). Note for a PCI Endpoint that supports Native IOV, mechanisms exist such that VI Run-Time Involvement may not be required.
SI - System Image	A software component running on a Virtual System to which specific virtual and physical Endpoints can be assigned. Specification of the behavior and architecture of an SI is outside the scope of this specification. Examples of SIs include guest operating systems and shared/non-shared protected domain Endpoint drivers.
Translated Address	An address formed by using the results of an address translation request. (Refer to the ATC specification for updates to the definition.)
Translation Agent (TA)	A logical entity that converts addresses expressed in terms of one address space into an address in a different address space. A Translation Agent (TA) is associated with memory that contains translation tables that the TA will reference for address translation (See ATPT). A TA may contain an Address Translation Cache. (Refer to the ATC specification for updates to the definition.)
Un-translated Address	An address that is formed using the existing programming mechanisms of PCI Express. (Refer to the ATC specification for updates to the definition.)
Guest Virtual Address	The address of a location in virtual memory that is assigned to a user application within an SI and, typically, is protected from access by other user applications. It is typically translated to a Guest Physical Address and then to a Physical Address. The processor's address translation mechanisms generally are used to implement such mappings.
VE - Virtual Endpoint	A part of an Endpoint that shares one or more physical Endpoint resources, such as a link, with another VE and supports Native Based

IOV.

.

1

1. Introduction

This chapter presents an overview of PCIe IO Virtualization and key concepts. PCI IO Virtualization is the capability for a single physical PCI IO Endpoint to be used by more than one System Images (SI) running in the same or on different PCIe Root Complex (RC).

Single RC topologies can be classified by the level of virtualization offered:

- Dedicated Endpoint - No virtualization support. A physical Endpoint is associated with only one System Image and cannot be shared between SI.
- Virtualization through the use of software or hardware intermediaries - Virtualization is provided through a software or hardware intermediary, which provides the mechanisms required to share the Endpoint, including: access controls and multiplexing/de-multiplexing of data movement, interrupt, and configuration operations. This level of virtualization does not require any additional functions in the physical PCIe Endpoint.
- Native IO Virtualization - Virtualization is provided by the physical PCI Endpoint. The physical PCI Endpoint supports one or more Virtual Endpoints (VE). Mechanisms exist such that each VE can, without run-time intervention by a Virtualization Intermediary (VI), directly: sink I/O and Memory operations from a System Image; and source DMA, completion, and interrupt operations to a System Image. A VI may still be required for configuration operations.

The above three levels of virtualization are also possible in a multi-root topology that interconnects multiple platforms, with their respective RC, to one or more multi-root capable Switches that support sharing Endpoints (the Endpoint may or may not be capable of native virtualization support).

The next sections of this specification describe different types or IO Virtualization. Subsequent chapters describe the semantics associated with Translation Agents, Native IO Virtualization, and multi-root topologies.

2

2. IO Virtualization Architectural Overview

This specification describes IO Virtualization (IOV) architecture for single-root and multi-root topologies. IOV for single-root topologies enables the native virtualization of enabled Endpoints for a single-root. IOV for multi-root topologies enables the native virtualization of enabled Endpoints for multi-root topologies.

Native based PCI IOV is a mechanism where a physical PCI Endpoint supports one or more Virtual Endpoint (VE). The configuration operations are performed using a virtual intermediary (VI) and the data movement and interrupt operations can be performed directly between the Operating System and the VE. A VE shares one or more physical Endpoint resources, such as a link, with another VE and can directly sink I/O and memory operations from a System Image (SI) and can source DMA, completion, and interrupt operations to an SI. Configuration operations flow through a Virtualization Intermediary.

A multi-root Topology is one where each root complex can support at least one PCI address space, separate from any other root complexes PCI address spaces.. Additionally, multiple root complexes may contain resources for a single shared Endpoint in their PCI address space. The configuration of these separate address spaces shall be managed by a single entity called the PCI Manager (PCIM). The PCIM shall be able to configure and discover IOV and non-IOV enabled RC's, Switches, Bridges and Endpoints. The PCIM shall support more than one PCIM instance for redundancy, and it shall support a mechanism to coordinate ownership of the IOV fabric among multiple PCIMs.

Different Types of Mechanisms for PCI Virtualization can be identified. These possible topologies and their attributes are described in the following sub-sections. Of these types of topologies, the ones that are covered by this specification are Native IOV for single and multi-root Topologies. More details about these two topologies are presented in subsequent chapters.

The scope or scalability of an IOV topology can vary as a function of the components used. For example, a single-root topology may consist of a set of RP with point-to-point connections to an IOV Endpoint while a multi-root topology may consist of multiple RC interconnect by a small number of multi-root aware Switches. As a guiding principle, IOV topologies are expected to be relatively small, i.e. within a single enclosure or perhaps only within a single rack. The IOV specification does not support large, arbitrary fabric topologies thus avoids the complex problems associated with such topologies.

2.1. Dedicated Endpoints

We should describe the operation flows and pros/cons associated with this topology.
A PCI topology or configuration where there are only Dedicated Endpoints, which do not use IO Virtualization. The platform and the Endpoints may or may not support PCIe IO Virtualization, but

if they do it is not turned on or in use. This topology is presented here for reference purposes and to use as a level set to which to compare other topologies that support different kinds of IO virtualization. Figure 1 illustrates a typical configuration with dedicated Endpoints where the System Image (SI 1) owns all the PCI IO resources.

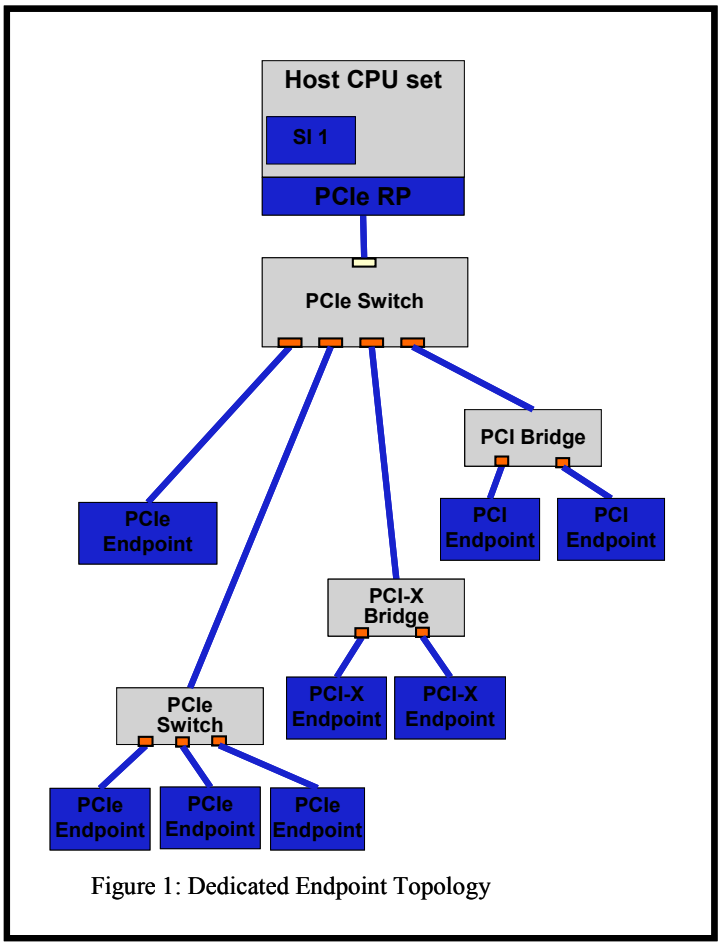


Figure 1: Dedicated Endpoint Topology

2.2. Endpoint Sharing Via Software Intermediary

We may want to combine this section and the next. We should describe the operation flows and pros/cons associated with this topology.

Another PCI topology has Endpoint Sharing through Software Intermediary. In this case the platform supports multiple SI. The IO Endpoints below the PCI Root Complex on the platform can be shared between the SI through a Software Intermediary. The intermediary can be a Hypervisor, a Virtual IO Server Partition, or other software mechanism. The programming model for a software intermediary is usually implemented at the Endpoint "class" level. (e.g. disk read, disk write, network read, network write, etc.) Figure 2 depicts a typical configuration with two SI in the

Platform CPU set. With the use of a software intermediary, the Endpoints below the PCIe Root Complex can be shared between SI1 and SI2.

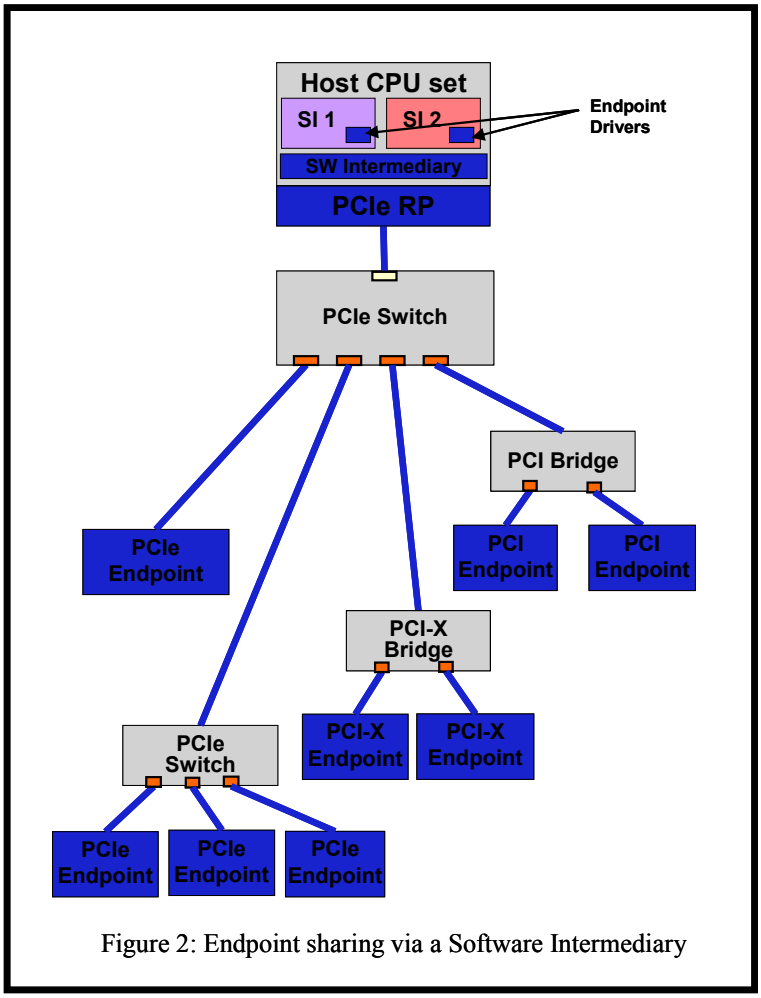


Figure 2: Endpoint sharing via a Software Intermediary

2.3. Address Translation and Protection Table

A platform or chipset may include an optional Address Translation and Protection Tables (ATPT). While ATPT are not standardized, many implementations will provide common basic services:

- Address translation – Translation of the address field within the TLP into a target address within the platform. Translation can occur for both DMA Read and DMA Write operations.

- Address protection – Access control of what Endpoint is allowed to access a given address. For example, a given address may be associated with one or more Requester ID extracted from a TLP. If the Requester ID is enabled to access the address, then the request is processed normally.
- The ATPT is programmed via software intermediary and not individual SI. This insures that one SI cannot erroneously (software or hardware fault) or intentionally access another SI. With an ATPI present, an SI can directly communicate without having to trap to an intermediary for every access. This can provide increased performance while improving SI isolation and protection.

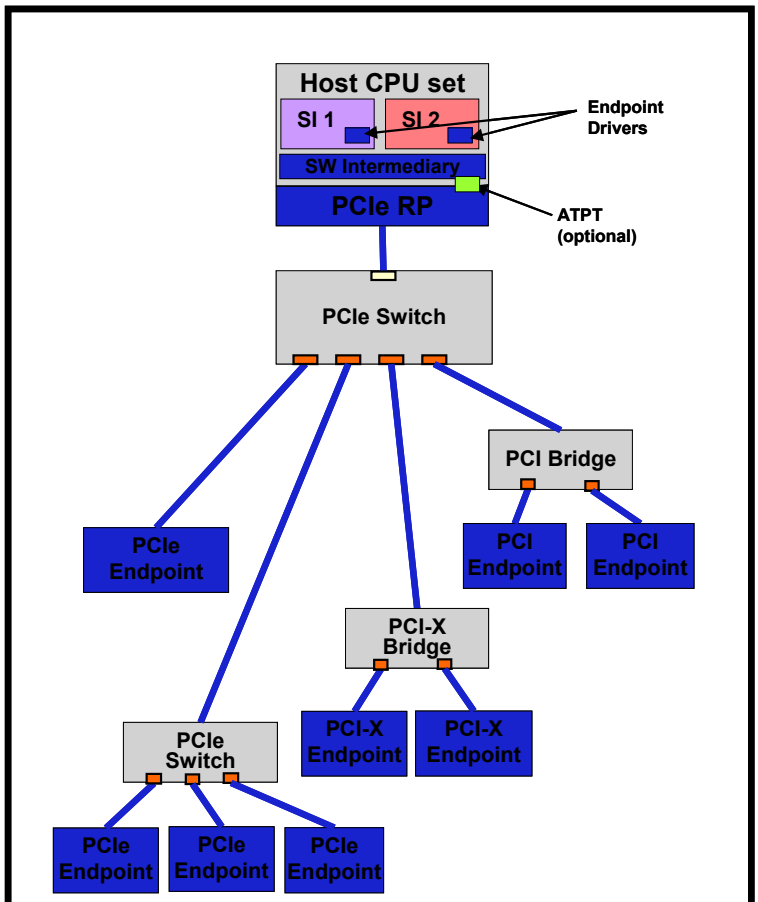


Figure 3: Endpoint Sharing via a Software Intermediary and with an optional ATPT Present

2.4. Native IOV

This section defines high-level specification requirements to support Native IO Endpoints. The primary objective of a Native IOV solution is to enable performance critical transactions to be processed without requiring direct interaction with a VI on a per transaction basis. Non-

performance critical transactions may require interaction with a VI, e.g. during configuration or error handling.

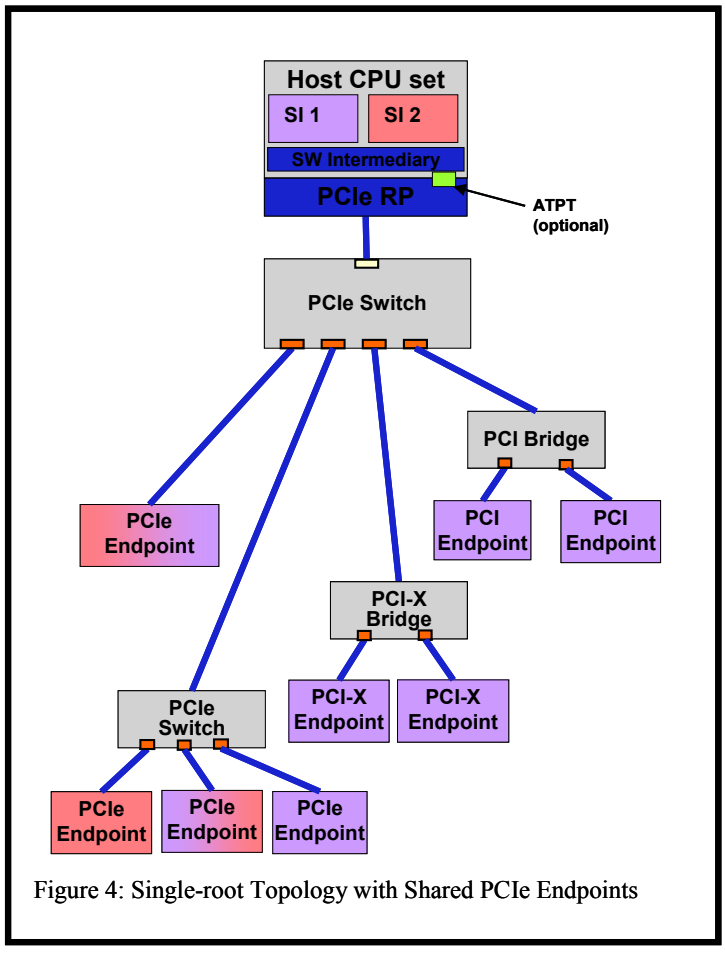


Figure 4: Single-root Topology with Shared PCIe Endpoints

The requirements for single-root Endpoints covered in this specification are described in the following text.

1) The only Endpoints that the PCI-SIG IOV WG shall actively specify for are PCIe Endpoints.
2) IOV Endpoints shall be backwards compatible, in a non IOV mode, with PCIe Base 1.1 specification.
3) Single-root IOV Endpoints shall be a superset of the PCIe Base 1.1 specification.

4) The impact or change requirements to existing enumeration and configuration algorithms shall be minimized.
5) Each VE within an IOV Endpoint shall have the capability to source DMA, completion, and interrupt operations within the system as an initiator without requiring an intermediary. DMA requests from each VE must be uniquely identifiable by the RC. Each VE must be capable of independently targeting its interrupts.
6) Each VE within an IOV Endpoint shall be able to process configuration, I/O, and memory operations within the system as a target. Each VE must have the ability to publish independent and distinct configuration, I/O and MMIO spaces. An SI shall be able to initiate memory operations directly to a VE without requiring an intermediary
7) Each VE of an IOV enabled Endpoint shall appear in configuration space as one or more PCIe function(s).
8) IOV Endpoints shall support a standard in-band mechanism allowing software to manage the IOV Endpoint.
9) IOV enabled Endpoints shall include functionality to disable their peer-to-peer traffic to other functions within the Endpoint.
10) IOV enabled Endpoints shall be allowed to support multiple functions assigned to the same SI with the same level of functionality given to a current multi-function Endpoint.
11) IOV enabled Endpoints shall support defined arbitration mechanisms between the VE within an IOV enabled Endpoint
12) Each Switch, Bridge, function, and VE shall be uniquely represented in the configuration space of the RC.
13) IOV enabled Endpoints shall support the capability to have each VE independently managed, including functional reset and be enabled/disabled. IOV enabled Endpoints shall support the capability to have a VE PCIe state and resources modified. Examples include: TC assignment, Bus Master Enable, etc.
14) IOV enabled Endpoints shall support the specification of the minimum and maximum number of fabric-related resources [TBD] available to each VE within the IOV enabled Endpoint. For example, the MMIO region assigned to SIs can vary in size between SIs.
15) Except for errors that occur within an IOV enabled Endpoint which cannot be isolated to a particular VE, each transaction issued by an IOV enabled Endpoint shall be uniquely identifiable with the VE that sourced the transaction.
16) Except for errors, an IOV enabled Endpoint shall be able to associate each transaction it sinks with the appropriate VE. An example of such errors might include a link CRC error that causes the request ID field integrity to be questionable.
17) An IOV Endpoint shall utilize a defined error mapping method to determine how errors are contained to and reported within a VE or to the PCIM
18) A method must be defined to identify VE-specific errors. This may be as simple as AER and include Requester ID, Completer ID, etc. to identify the VE.
19) Error isolation needs to delineate between errors that impact all VE within a given Endpoint, e.g. a link failure, vs. errors that only impact a subset of the VE.

20) A mechanism must be provided that allows disabling of peer-to-peer traffic between PCI components assigned to different SI.

21) A mechanism shall be provided to allow a VE to be associated with a System Image, such that data movement operations are enabled and can be performed directly between a System Image and a VE associated with that System Image, without VI involvement.

22) The virtualization mechanisms defined in this specification may require a Virtualization Intermediary to be involved for configuration operations performed on a VE.

23) Physical hot-plug/hot-removal of a physical Endpoint, Switch, or Bridge shall be supported. A VE shall be allowed to be added or removed dynamically.

24) An IOV enabled Endpoint shall use MSI-X.

 a. Question for membership: Can we eliminate INTx support entirely?

2.5. Multi-Root Topology

The previous subsections presented the levels or PCI IO Virtualization that are possible using different mechanisms, and were presented in physical configurations that only had a single RC with one or several SI. This subsection presents a generalized configuration with more than one platform and its associated PCIe Root Complexes connected to the same Switch. The multiple RC, each potentially having multiple SI, can share the same set of Endpoints. This topology presents extra requirements and these will be presented in the second half of this PCIe IO Virtualization specification. Figure 6 shows a typical multi-root topology.

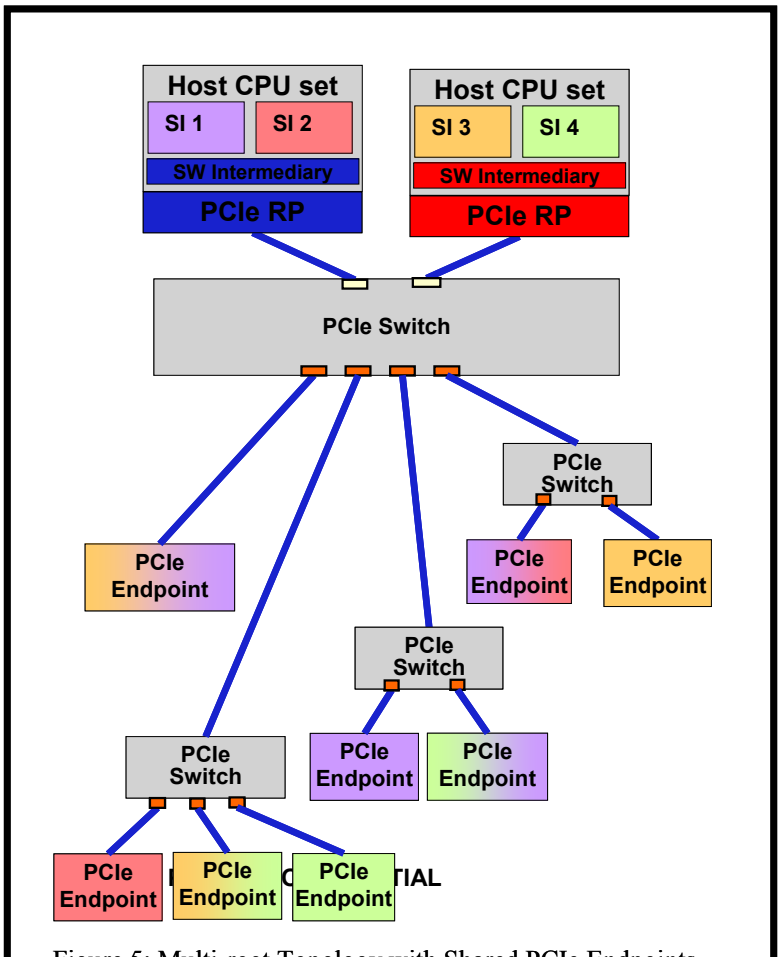


Figure 5: Multi-root Topology with Shared PCIe Endpoints

Of the configurations and topologies presented above, this specification covers the requirements for Native IO Virtualization Endpoints in the following chapters (Chapter 3 to Chapter 6) and of multi-root topologies in Chapters Y to Z.

The requirements for multi-root topologies covered in this specification are described in the following text.

1) The multi-root solution shall provide the same characteristics to its IOV enabled Endpoints as the single-root solution relative to separate SI.
2) The multi-root solution shall give each RC its own Virtual Hierarchy.
3) The multi-root solution shall enable use of existing PCIe 1.1 or later RC.
4) The multi-root solution shall enable existing PCIe 1.1 Switches, Endpoints, and PCIe to PCI/PCI-X Bridges to each be assigned to a single RC.
5) The multi-root solution shall enable multiple VEs from an IOV Endpoint to be shared amongst multiple roots using a MRA Switch.
6) The multi-root solution shall enable a MRA Bridge to PCI/PCI-X to assign each bridge function to a separate root.
7) The multi-root solution shall enable multiple MRA Switches to be connected such that multiple single rooted hierarchy can be interconnected, providing that the interconnected hierarchies appear to be a hierarchy structure to each RC.
8) The multi-root solution shall not make special provisions to resolve loops created by connecting multiple MRA Switches together.
9) The multi-root solution shall enable an RC to exist on any upstream Port of any MRA Switch.
10) The multi-root solution may enable a mechanism to manage, limit, or contain the congestion caused by one RC as that congestion pertains to other RC's in the system
11) The multi-root solution should enable software written for PCI, PCI-X, or PCIe enumeration, configuration, and management to run, unchanged on each RC within the multi-root system
12) The multi-root solution shall enable a MRA Endpoint to be shared by multiple SI within one or more RCs in a multi-root fabric.
13) Each Switch, Bridge, function, and VE shall be uniquely represented in the configuration space of each RC.

14) The multi-root solution shall enable, when possible, isolation of a PCIe failure of a given RC to only components solely within the Virtual Hierarchy associated with that RC.

15) The multi-root solution shall enable through an MRA Switch isolation of a PCIe failure within a Virtual Hierarchy associated with a given RC to ensure that virtual components in other Virtual Hierarchies are not affected.

16) The multi-root solution shall enable a management entity (PCIM) to manage the fabric.

17) Enabling RC-to-RC data transfer or other communications is a non-requirement for the multi-root solutions.

18) Multi-root capabilities shall be a superset of the single-root capabilities. The objective is to maximize component interoperability across multiple topologies.

19) An Endpoint, Switch, or RC component designed to support multi-root **shall** support the single-root capabilities.

20) Multi-root topologies can be built using single-root IOV Endpoints and RC. Non MRA components will be limited to a single virtual hierarchy. An MRA Switch is required to interconnect these components.

21) The multi-root solution shall support physical hot-plug/hot-removal of a physical PCI Endpoint, Switch, or Bridge. Additionally, virtual hot-plug/hot-removal of a VE or a Switch Port from a Virtual Hierarchy shall be supported.

22) Peer-to-peer traffic between PCI components in different virtual hierarchies shall not be allowed.

2.6. MRA PCI Manager Operating Principles

Multi-root configuration and management requirements are driven by the basic operating principles of a PCI Manager (PCIM). These operating principles are described as follows:

1) The PCIM is able to discover and configure IOV enabled RC, Switches, Bridges, and Endpoints using a defined PCIM interface.

2) The PCIM is able to discover and configure non-IOV components using existing interfaces.

3) A solution may support one or more PCIM entities. For implementations that support more than a single PCIM, each PCIM entity is capable of coordinating with all other PCIM entities.

4) The PCIM is able to control and manage errors within the fabric.

5) The PCIM is able to establish a trust relationship between itself and a given PCIe component.

6) A PCIM is able to control and limit access to IOV management functions within IOV enabled components by other entities, e.g. a secondary PCIM or PCI firmware associated with only a single root.

7) The PCIe multi-RC configuration provides multiple configuration spaces for each switch, bridge and Endpoint.

3

3. Native IOV – Protocol Changes

The methods will be defined once the requirements are finalized.

4

4. Native IOV – IOV Semantics

The methods will be defined once the requirements are finalized.

5

5. Native IOV Initialization / Configuration

5.1. Configuration Mechanisms

The PCIe Native IO Virtualization Configuration model supports the following configuration space access mechanisms:

◊ PCI compatible configuration mechanism

◊ PCIe enhanced configuration mechanism

◊ Native IOV (within PCIe enhanced configuration) mechanism

The PCI compatible and PCIe enhanced mechanisms have the same semantics as described in the Release 1.1 of the PCIe Specification. The Native IOV mechanism uses fields within the PCIe enhanced configuration mechanism to:

◊ Determine how many VE are supported by the Endpoint

◊ Select the mode of operation for the Endpoint:

- Base Mode - In this mode, the Endpoint is configured as a single Endpoint and cannot be configured as multiple VE. This is the default mode and allows the Endpoint to be used in PCIe topologies that do not support Native IOV.
- Virtual Entity Mode - In this mode, the Endpoint is configured to enable multiple VE. This mode is intended for use on a PCIe topology that supports Native IOV.

◊ Allocate a Virtual Entity within the Endpoint

◊ Destroy a Virtual Entity within the Endpoint

Note: Defining configuration and management policy issues is a non-requirement. The mechanisms used to define and drive policy based decisions (such as how to configure a native IOV enabled Endpoint, how many VE to create, which SI/RC the VE are assigned to, etc) are beyond the scope of this specification.

5.2. Configuration Topology

To maintain compatibility with PCIe software configuration mechanisms, all PCIe Endpoints that enable Native IOV have a PCI-compatible configuration space representation.

Native IOV does not require any changes to a PCIe Release 1.1 Root Complex. Therefore, the same Endpoint Mapping mechanisms used by a PCIe Release 1.1 Root Complex are used when Native IOV is enabled on a PCIe Endpoint.

Native IOV does not require any changes to a PCIe Release 1.1 Switch. Therefore, the same Endpoint Mapping mechanisms used by a PCIe Release 1.1 Switch are used when Native IOV is enabled on a PCIe Endpoint.

A PCIe Endpoint that supports Native IOV has two possible configurations: one associated with the Physical Mode of operation; and the other with the Virtual Entity Mode of operation.

This section will describe each of these configurations.

5.3. Configuration Mechanisms

Native IOV does not require any changes to a PCIe Base Specification Release 1.1 RC. Therefore, the same Configuration mechanisms used by a PCIe Base Specification Release 1.1 RC are used when Native IOV is enabled on a PCIe Endpoint.

Native IOV does not require any changes to a PCIe Base Specification Release 1.1 Switch. Therefore, the same Configuration mechanisms used by a PCIe Base Specification Release 1.1 Switch are used when Native IOV is enabled on a PCIe Endpoint.

This section will describe the configuration mechanisms used to select the mode of operation for a Native IOV enabled PCIe Endpoint.

5.4. Configuration Transaction Rules

Native IOV does not require any changes to a PCIe Base Specification Release 1.1 Root Complex. Therefore, the same Configuration Transaction Rules applicable to a PCIe Base Specification Release 1.1 Root Complex apply when Native IOV is enabled on a PCIe Endpoint.

Native IOV does not require any changes to a PCIe Base Specification Release 1.1 Switch. Therefore, the same Configuration Transaction Rules applicable to a PCIe Base Specification Release 1.1 Switch apply when Native IOV is enabled on a PCIe Endpoint.

This section will describe the configuration transactions rules for a Native IOV enabled PCIe Endpoint.

5.5. Configuration Space

Native IOV does not require any changes to a PCIe Base Specification Release 1.1 Root Complex. Therefore, the same Configuration Register Types applicable to a PCIe Base Specification Release 1.1 RC apply when Native IOV is enabled on a PCIe Endpoint.

Native IOV does not require any changes to a PCIe Base Specification Release 1.1 Switch. Therefore, the same Configuration Register Types applicable to a PCIe Base Specification Release 1.1 Switch apply when Native IOV is enabled on a PCIe Endpoint.

This section will describe the configuration space used in each mode of operation for a Native IOV enabled PCIe Endpoint.

5.6. Configuration Space Structure - Physical Mode

All the syntax and semantics described in the PCIe Base Specification Release 1.1 Specification section 5.5 to 5.13 apply as is to a PCIe Endpoint that is enabled for Native IOV and is in the Physical Mode of operation.

5.7. Configuration Space Structure – VE Mode

This section will describe how the syntax and semantics of the Configuration Space fields described in section 5.5 to 5.13 of the PCIe Base Specification Release 1.1 Specification apply when a PCIe Endpoint is in the VE Mode of operation.

5.8. VE Creation

This section will describe how the syntax and semantic associated with VE Creation.

5.9. VE Modification

This section will describe how the syntax and semantic associated with VE Modification.

5.10. VE Destruction

This section will describe how the syntax and semantic associated with VE Destruction.

6

6. Native IOV - Event Management

This section will describe the various types of events, such as VE hot-plug, VE hot-remove, etc…

6.1. VE Reset

This section will describe how the syntax and semantic associated with VE Reset.

6.2. VE Hot-Plug

This section will describe how the syntax and semantic associated with VE Plug.

6.3. VE Hot-Remove

This section will describe how the syntax and semantic associated with VE Remove.

7

7. Native IOV – Error Reporting

The methods will be defined once the requirements are finalized.

7.1. Error Reporting Mechanisms

The methods will be defined once the requirements are finalized.

8

8. Native IOV – Power Management

The methods will be defined once the requirements are finalized.

9

9.Multi-Root IOV Semantics

The methods will be defined once the requirements are finalized.

10

10. Multi-Root Protocol Changes

This section will be removed in the final version.

The following protocol changes are required to support multi-root IO Virtualization:

◊ Transaction Layer

- changes to support multiple memory, I/O and configuration address spaces, at least one set per PCIe RC
- changes to separate message routing by PCIe RC involved
- changes to manage INTx processing by PCIe RC
- changes to allow hot plug and power management to be virtualized independently to each PCIe RC

◊ Data Link Layer

- changes to support propagation of a per-RC version of Hot Reset
- changes to support auto-negotiation of shared link mode

◊ Physical Layer - None.

11

11. Multi-Root Initialization / Configuration

PCIe multi-root topologies utilize multiple configuration spaces for each Switch, Bridge and Endpoint.

Each Virtual Entity and Physical Entity supports the following configuration space access mechanisms:

◊ PC compatible and PCIe enhanced configuration mechanism

These have the same semantics as described in Release 1.1 of the PCIe Base Specification.

Each Virtual Entity has configuration space that allows software running on a PCIe Root Complex full control of the Virtual Entity as if it were a PCIe Base Specification 1.1 physical entity.

Each Physical Entity has configuration space to provide an access path for the PCIM software to the Native IOV configuration mechanism.

◊ Native IOV (within PCIe enhanced configuration) mechanism.

Each Virtual Entity has an optional configuration space providing a communication path between software running on a PCIe RC and the PCIM software.

Each Physical Entity has configuration space that allows the PCIM software to manage the Virtual Entities. Using this mechanism, the PCIM can perform tasks such as: enable and disable Virtual Entities, manage resource allocation between Virtual Entities, communicate with software using the Virtual Entity optional communication path, etc.

11.1. Configuration Topology

This section will describe the supported IOV fabric topologies and how those topologies are visible to configuration and management software.

11.2. Configuration Mechanisms

This section will describe the overall mechanisms used to by a PCIM to manage an IOV fabric. This will include a discussion of the PCIM visible entities and what role those entities perform in managing the fabric.

11.3. Configuration Transaction Rules

This section will describe the rules used by PCIM in enumerating and configuring an IOV fabric. This will include the ordering requirements placed on PCIM software.

11.4. Configuration Space

This section will describe the configuration space registers used by the PCIM to manage the IOV fabric. It will also describe registers used by IOV enabled software to interact with the PCIM software.

11.5. Trusted Configuration Space

11.6. Electing a PCIM to Manage the IOV Fabric

This section will describe the hardware used to allow multiple PCIM software entities to choose amongst themselves which one is in control of the fabric and to configure the fabric to accept the chosen manager. The standard will NOT address the policy side of making the choice nor will it specify how software uses the provided hardware (indeed PCIM software may ignore the hardware and use other mechanisms)

11.7. Enforcing PCIM Election

This section will describe how the results of a PCIM election are enforced. This enforcement allows the chosen PCIM to manage the IOV fabric while prohibiting other entities from doing so.

11.8. Transfer of PCIM Election

This section will describe how management authority is transferred from one PCIM to another PCIM. It will describe both planned and unplanned transfer. It will also discuss the mechanism used to detect PCIM failure that triggers an unplanned transfer of authority.

11.9. Establish an Error Collector

This section will describe how PCIM software configures the IOV enabled entities to send certain error messages to PCIM software. The errors involved are described in a subsequent chapter.

11.10. Association of Virtual Entity to an RC

This section will describe how PCIM software maps a Virtual Entity of an Endpoint into the hierarchy of an RC. .

11.11. Reset of an Endpoint associated with an RC

This section will describe how PCIM software resets a Virtual Entity associated with an RC.

11.12. Disassociation of an Endpoint from an RC

This section will describe how PCIM software unmaps a Virtual Entity from the hierarchy of an RC and makes the Virtual Entity available for subsequent mapping into some RC (either a different RC or back to the same one).

11.13. Monitoring an IOV Fabric

This section will describe how a PCIM collects statistics, counters, congestion information, etc. from an IOV fabric.

12

12. Native IOV - Event Management

This section will describe the various types of events, such as VE hot-plug, VE hot-remove, etc… The methods will be defined once the requirements are finalized.

12.1. MRA Endpoint Reset

This section will describe how the syntax and semantic associated with VE Reset.

12.2. MRA Endpoint Hot-Plug

This section will describe how the syntax and semantic associated with VE Hot-plug.

12.3. MRA Endpoint Hot-Remove

This section will describe how the syntax and semantic associated with VE Remove.

This section will describe how the syntax and semantic associated with VE Deletion

13

13. Multi-Root- Error Reporting

The methods will be defined once the requirements are finalized.

13.1. Error Reporting Mechanisms

14

14. Multi-Root - Power Management

The methods will be defined once the requirements are finalized.